Node

1. **Create http server**

```
const http = require("http");
const fs = require("fs");
const myServer = http.createServer((req,res) => {
    const log = `${Date.now()}: ${req.url} New Req Received\n`;
    fs.appendFile("log.txt",log,(err,data)=>{
        switch(req.url){
            case "/":
                res.end("Homepage");
                break;
            case "/about":
                res.end("I am Sithkar");
                break;
            case "/contact":
                res.end("Say Hii to me");
            default:
                res.end("404 Page Not Found");
        }
    });
});
myServer.listen(8000,() => console.log("Server Started"));
```

2. Handling URL(Uniform Resource Locator)

```
const http = require("http");

const fs = require("fs");

const url = require("url");

const myServer = http.createServer((req,res) => {

    if(req.url === "/favicon.ico") return res.end();

    const log = `${Date.now()}: ${req.url} New Req Received\n`;

    const myUrl = url.parse(req.url,true);

    console.log(myUrl);

    fs.appendFile("log.txt",log,(err,data)=>{

        switch(myUrl.pathname){

            case "/":

                res.end("Homepage");

                break;

            case "/about":

                const username = myUrl.query.myname;

                res.end(`Hi, ${username}`);

                break;

            case "/search":

                const search = myUrl.query.search_query;

                res.end("Here are your results for " + search);

            default:

                res.end("404 Page Not Found");

        }

    });

});
```

```
myServer.listen(8000,() => console.log("Server Started"));
```

3. HTTP Method (Get, Post, Put , Patch , Delete)

```
const http = require("http");
const fs = require("fs");
const url = require("url");
const myServer = http.createServer((req,res) => {
    if(req.url === "/favicon.ico") return res.end();
    const log = `${Date.now()}:${req.method} ${req.url} New Req Received\n`;
    const myUrl = url.parse(req.url,true);
    console.log(myUrl);
    fs.appendFile("log.txt",log,(err,data)=>{
        switch(myUrl.pathname){
            case "/":
                if(req.method === "GET"){
                    res.end("Homepage");
                }
                break;
            case "/about":
                const username = myUrl.query.myname;
                res.end(`Hi, ${username}`);
                break;
            case "/search":
                const search = myUrl.query.search_query;
                res.end("Here are your results for " + search);
            case "/signup":
                if(req.method === "GET") res.end("This is a signup form");
                else if(req.method === "POST"){
                    // DB Query
                    res.end("Success");
                }
            default:
                res.end("404 Page Not Found");
        }
    });
});
myServer.listen(8000,() => console.log("Server Started"));
```

4. Express (framework)

```
const express = require("express");
const app = express();
```

```
app.get("/",(req,res)=>{
    return res.send("Hello from ");
})
app.get("/about",(req,res) =>{
    return res.send("Hello from about page");
})
app.listen(8000, () => console.log("Server Started!"));
```

Version

```
// Version
4.18.3
1st Part -> 4
2nd Part -> 18
3rd Part -> 3
// 3rd Part(Last Part) -> Minor Fixes(Optional)
// 2nd Part -> Recommended Bug Fix (Secure)
// 1st Part Major Release -> Major / Breaking Update
^ -> Install all Recommended and Minor Fixes Automatic
~ -> Only last wala change hua thabhi update krega
```

REST API

- Work on server-client archietecture
- JSON -> JavaScript Object Notation (client side rendering)
- If you know that your client is Browser then we send HTML format (Server Side Rendering) because it's fast than JSON
- Always respect all http methods

```
GET /user -> read the user data and return the data
POST /user -> handle new user creation
PATCH /user -> update the user
To Generate the fake JSON data use -> mockaroo.com
```

- **Creating REST API**

```
const express = require("express");
const users = require("./MOCK_DATA.json");
const app = express();
const PORT = 8000;
// Routes
app.get('/api/users',(req,res) => {
    return res.json(users);
});
app.route("api/users/:id")
    .get((req,res) => {
    const id = Number(req.params.id);
    const user = users.find((user) => user.id === id);
```

```
    return res.json(user);
})
.patch((req,res) => {
    // TODO : Edit the user with id
    return res.json({status : "Pending"})
})
.delete('/api/users/:id',(req,res) =>{
    // TODO : Delete the user with id
    return res.json({status:"pending"});
});
app.post('/api/users',(req,res) => {
    // TODO : Create new user
    return res.json({status:"pending"});
});
app.listen(PORT,() => console.log(`Server Started at PORT`))
```

- **HTTP Headers**

HTTP Headers are an important part of the API request and response as they represent the meta-data associated with the API request and response.

Headers carry information for the request and response body.

- **Status Code**

404 – Not Found

Informational responses : 100 - 199

Successful responses : 200 - 299

Redirection messages : 300 - 399

Client error responses : 400 - 499

Server error responses : 500 – 599

- **MiddleWare**
- const express = require("express");
- const fs = require("fs");
- const mongoose = require("mongoose");
- const users = require("./MOCK_DATA.json");
- const app = express();
- const PORT = 8000;
- // Middleware - Plugin
- app.use(express.urlencoded({ extended: false}));
- // app.use((req,res,next) =>{
- //   console.log("Hello from middleware 1");
- //   // return res.json({msg: "Hello from middleware 1"});
- //   next();
- // });
- // app.use((req,res,next) =>{
- //   console.log("Hello from middleware 2");
- //   return res.end("Hey");
- // });
- // Routes
- app.get('/api/users',(req,res) => {
-    return res.json(users);
- });
-
- app.route("api/users/:id")
-     .get((req,res) => {

- const id = Number(req.params.id);
- const user = users.find((user) => user.id === id);
- return res.json(user);
- })
- .patch((req,res) => {
- // TODO : Edit the user with id
- return res.json({status : "Pending"})
- })
- .delete((req,res) =>{
- // TODO : Delete the user with id
- return res.json({status:"pending"});
- });
- app.post('/api/users',(req,res) => {
- // TODO : Create new user
- const body = req.body;
- users.push({...body,id: users.length +1 });
- fs.writeFile("./MOCK_DATA.json",JSON.stringify(users),(err,data) => {
- return res.status(201).json({status: "success",id:users.length})
- });
- });
- app.listen(PORT,() => console.log(`Server Started at PORT`))
- **MongoDB**
- **No-SQL Document based Database**
- **Strong support for Aggregation Pipes**
- **Works on BSON format**
- **Best for Node Applications**

Coder Dost

3 hour

```
const fs = require('fs');

const index = fs.readFileSync('index.html','utf-8');

const data = JSON.parse(fs.readFileSync('data.json','utf-8'));

const products = data.products;

const express = require('express');

const morgan = require('morgan');

const { type } = require('os');

const server = express();

//bodyparser

//server.use(express.json());

// server.use(morgan('default'))

// server.use((req,res,next)=>{

//     console.log(req.method,req.ip,req.hostname,new Date());

//     next();

// })

//MiddleWare

//const auth = (req,res,next) =>{

   // console.log(req.query);

    // if(req.body.password=='1234'){

    //     next();

    // }

    // else{
```

```
    //     res.sendStatus(401);
    // }
//     next();
// }
// server.use(auth);
//API - EndPoint Routes
//API ROOT, basee URL ,google.com/api/v2/
server.get('/products',(req,res)=>{
    res.json(products);
});
// Read GET /products/:id
server.get('/products/:id',(req,res) =>{
    const id = +req.params.id;
    const product = products.find(p=>p.id===id);
    res.json(product);
});

server.get('/product/:id',(req,res)=>{
    console.log(req.params);
    res.json({type:'GET1'});
})
server.get('/',(req,res)=>{
    res.json({type:'GET1'});
})
server.post('/',(req,res)=>{
    res.json({type:'POST'});
})
// server.patch('/',auth,(req,res)=>{
//     res.json({type:'PATCH'});
// })
server.put('/',(req,res)=>{
    res.json({type:'PUT'});
})
server.delete('/',(req,res)=>{
    res.json({type:'DELETE'});
})
server.get('/',(req,res)=>{
    // res.send('<h1>hello<h1/>')
    res.json(product);
```

```
})


server.listen(8080,()=>{
    console.log('server started')
});
```

3:33

CURD Operation

```
const fs = require('fs');
const index = fs.readFileSync('index.html','utf-8');
const data = JSON.parse(fs.readFileSync('data.json','utf-8'));
const products = data.products;
const express = require('express');
const morgan = require('morgan');
const { type } = require('os');
const server = express();
//bodyparser
server.use(express.json());
//server.use(morgan('default'))
// server.use((req,res,next)=>{
//     console.log(req.method,req.ip,req.hostname,new Date());
//     next();
// })
//MiddleWare
//const auth = (req,res,next) =>{
   // console.log(req.query);
    // if(req.body.password=='1234'){
    //     next();
    // }
    // else{
    //     res.sendStatus(401);
    // }
//     next();
// }
// server.use(auth);
//API - EndPoint Routes
//API ROOT, basee URL ,google.com/api/v2/
//Create POST /products  C R U D
server.post('/products',(req,res) =>{
    console.log(req.body);
```

```
    products.push(req.body);
    res.status(201).json(req.body);
});
// Read GET/ products
server.get('/products',(req,res)=>{
    res.json(products);
});
// Read GET /products/:id
server.get('/products/:id',(req,res) =>{
    const id = +req.params.id;
    const product = products.find(p=>p.id===id);
    res.json(product);
});
// Update PUT /products/:id
server.put('/products/:id',(req,res) =>{
    const id = +req.params.id;
    const productIndex = products.findIndex(p=>p.id===id);
    products.splice(productIndex,1,{...req.body,id:id})
    res.status(202).json({product:'updated'});
});
// Update PATCH /products/:id
server.patch('/products/:id',(req,res) =>{
    const id = +req.params.id;
    const productIndex = products.findIndex(p=>p.id===id);
    const product = products[productIndex];
    products.splice(productIndex,1,{...product,...req.body})
    res.status(202).json({product:'updated'});
});
// DELETE /products/:id
server.delete('/products/:id',(req,res) =>{
    const id = +req.params.id;
    const productIndex = products.findIndex(p=>p.id===id);
    const product = products[productIndex]
    products.splice(productIndex,1)
    res.status(202).json(product);
});



server.listen(8080,()=>{
```

```
    console.log('server started')

});
```